



## Vantage RuntimeAI

Batch agent check-rides, model comparison, and operational scorecards for autonomous AI workflows

# 8.4/10

Overall score

### Run summary

Scenario: Train/Serve Skew

Model: mistralai/mistral-nemo

Roles: ML Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 1m 49s

Actual run cost: \$0.000234

Projected execution cost: \$0.0257/mo (5/biz day × 22 days)

Report date: July 13, 2026 - 11:48 PM EST

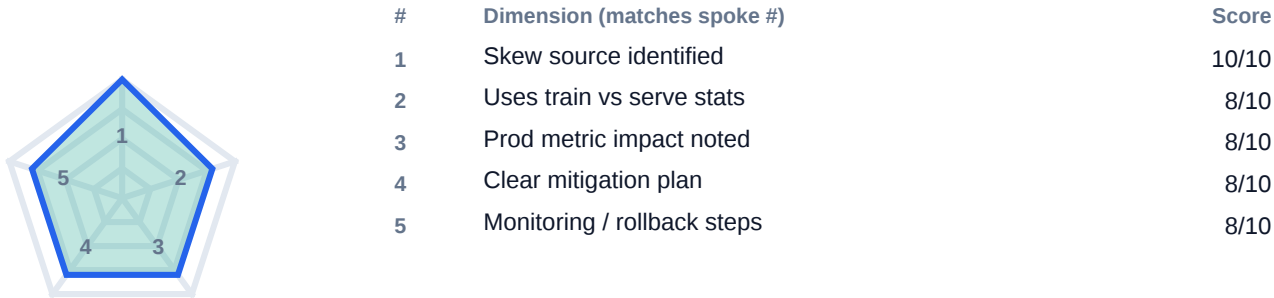
Overall

8.4/10

PASS - trust=high

Pass - score 8.4/10 clears 7.0 with acceptable trust/closure.

CAPABILITY RUBRIC



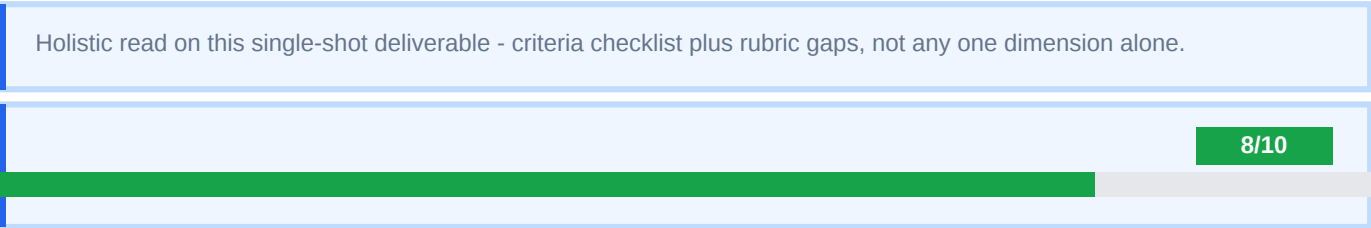
Spoke numbers run clockwise from the top; dimension names are listed in full in the table.

Technical criteria checklist

Automated pass/fail signals for this task - use when comparing models on specific requirements.

- Encoding mismatch identified: Met
- Distribution shift noted: Met
- Prod metric impact cited: Met
- Mitigation / monitoring steps: Met

TECHNICAL JUDGEMENT



Telemetry

## Run metrics

Mode: multi-turn - Agent msgs: 6 - Counterpart msgs: 7 - Output: 20957 chars - Latency: 109s - Criteria: 4/4

## Score analysis

**Identifies one-hot vs hash encoding mismatch as skew source.**

### Score snapshot

Solid: Solid technical task response with a few gaps.

### Interpretation

Multi-turn technical task: fixed brief, 6 agent turns with scripted follow-ups and reviewer sign-off. 4 of 4 checklist criteria passed.

### BRIGHT SPOTS (RUBRIC)

- + Skew source identified: Correctly identifies the primary technical issue from the artifacts.
- + Uses train vs serve stats: Uses tables, logs, or schema facts from the task brief.
- + Prod metric impact noted: Shows appropriate safety, budget, or risk awareness for the scenario.
- + Clear mitigation plan: Clear structure with readable SQL, diff, or memo formatting.
- + Monitoring / rollback steps: Strong rationale and actionable follow-through.

### DIMENSION NOTES

- Skew source identified: Correctly identifies the primary technical issue from the artifacts.
- Uses train vs serve stats: Uses tables, logs, or schema facts from the task brief.
- Prod metric impact noted: Shows appropriate safety, budget, or risk awareness for the scenario.
- Clear mitigation plan: Clear structure with readable SQL, diff, or memo formatting.
- Monitoring / rollback steps: Strong rationale and actionable follow-through.
- Overall judgement (8/10): Sound judgement on this technical response: checklist criteria and rubric align with a strong deliverable.

### Signals

Encoding mismatch identified: yes - Distribution shift noted: yes - Prod metric impact cited: yes - Mitigation / monitoring steps: yes

### CAVEATS

- Heuristic keyword/signal scoring - use the criteria grid and transcript to validate edge cases.

## Transcript

### Session

Scenario: Train/Serve Skew

Agent response <-> Engineering counterpart

ML Engineer (AI agent) <> Engineering counterpart (AI agent)

Duration: 1m 49s

## Engineering counterpart

3:46 AM

Task brief - Train/serve skew (complex complexity)

Production precision dropped after an encoding change. Investigate using the artifact:

```
```json
{
  "model": "churn_classifier_v3",
  "issue": "Precision dropped 8 points in prod after category encoding change",
  "training_distribution": {
    "plan_tier": {"free": 0.42, "pro": 0.38, "enterprise": 0.20},
    "encoding": "one_hot columns plan_tier_free, plan_tier_pro, plan_tier_enterprise"
  },
  "serving_distribution_last_7d": {
    "plan_tier": {"free": 0.61, "pro": 0.29, "enterprise": 0.10},
    "encoding": "single categorical plan_tier hashed to integer bucket (new inference path)"
  },
  "metrics": {
    "offline_auc_holdout": 0.84,
    "online_precision_7d": 0.71,
    "online_precision_baseline_30d": 0.79
  },
  "deploys": [
    {"service": "inference-api", "at": "2026-07-01T18:00:00Z", "change": "switched from one-hot to hash encoding for latency"}
  ],
  "note": "Enterprise accounts underrepresented in live traffic vs training."
}
```
```

## Agent response

3:46 AM - Responded in 9s

Skew Source (Encoding / Distribution):

1. Encoding Change: The encoding of the 'plan\_tier' feature was changed from one-hot to hashed during deployment. This change was made to improve inference latency, but it might have caused the precision drop in production.
2. Distribution Shift: There's a significant shift in the distribution of the 'plan\_tier' feature between the training data and the serving data over the last 7 days. In the training data, the 'free' tier accounts for 42% of the data, while in the serving data, they account for 61%. Conversely, 'enterprise' tier accounts for 20% in training and only 10% in serving. This underrepresentation of 'enterprise' tier in live traffic is a notable skew.

Mitigation Options:

1. Revert to Original Encoding (one-hot): If feasible, revert the encoding of 'plan\_tier' back to one-hot in the production model. This will bring the serving distribution closer to the training distribution.
2. Oversample 'enterprise' tier in Training: To address the underrepresentation issue, oversample or undersample 'enterprise' tier data during training to match the serving distribution. This can be done using techniques like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling).
3. sins(encoding): Use an encoding method that can handle skewness better, like target encoding or WOE (Weight of Evidence) encoding, which can capture non-linear relationships and handle skewness.
4. Feature Engineering: Create new features that might help the model generalize better to the new serving distribution, e.g., the ratio of 'free' to 'pro' tier users.

Monitoring and Rollback Recommendations:

1. Monitor Metrics: Keep a close eye on the online precision metric in production, and compare it with the baseline from 30 days ago. If the precision continues to drop or fluctuates, consider rolling back the encoding change.
2. Version Control and A/B Testing: Implement version control for your models and use A/B testing to compare the performance of the old and new encoding methods. This will help you make informed decisions and revert changes if necessary.
3. Shift Detection: Implement shift detection algorithms to monitor distribution shifts between training and serving data. This will help you identify similar issues early and take corrective actions.
4. Retry Old Encoding (canary release): As a temporary measure, release a new version of the service that uses the old one-hot encoding for the 'plan\_tier' feature alongside the current version. This will allow you to collect data on the performance of the old encoding in the new serving distribution and make a more informed decision about whether to roll back permanently.
5. Automated Rollback: If the precision drop is causing significant business impact, consider implementing an automated rollback system that reverses the encoding change if the online precision falls below a predetermined threshold.

## Engineering counterpart

3:46 AM - Responded in 0s

Thanks - walk me through your reasoning step by step. What assumptions did you make?

## Agent response

3:46 AM - Responded in 16s

I'm glad you found the initial response helpful! Let's break down the reasoning step by step and discuss the assumptions made.

### 1. Identify the issue and relevant factors:

- The problem: Production precision dropped after an encoding change.
- Relevant factors:
  - Encoding change: 'plan\_tier' feature was encoded differently (one-hot to hashed) in the deployed model.
  - Distribution shift: The distribution of 'plan\_tier' in serving data differs significantly from the training data, with 'free' tier being more prevalent and 'enterprise' tier being underrepresented in live traffic.

### 2. Assumptions made:

- Assumption 1: The encoding change and the distribution shift are the primary causes of the precision drop.
  - Rationale: The encoding change was the only mentioned model-related change, and distribution shift is a common cause of model performance degradation in real-world scenarios.
- Assumption 2: The 'plan\_tier' feature is an important feature for the model's prediction task.
  - Rationale: The observed distribution shift and encoding change involve this feature, suggesting its importance. Also, the note mentions that enterprise accounts are underrepresented in live traffic, which might be relevant to the model's performance.
- Assumption 3: The other features and the model architecture remain largely unchanged.
  - Rationale: The artifact doesn't mention any other model-related changes, so we can assume that the other features and the model architecture are relatively stable.

### 3. Analyze the impact of the encoding change:

- One-hot encoding: Represents each category as a separate binary feature, preserving the natural distribution of categories.
- Hashed encoding: Reduces the number of features by mapping several categories into a single bucket, potentially causing information loss, especially if the distribution of categories differs significantly between training and serving data.

### 4. Analyze the impact of the distribution shift:

- Underrepresentation of 'enterprise' tier: If the 'enterprise' tier has unique patterns that predict churn, the model might struggle to generalize to this segment due to its underrepresentation in live traffic.

### 5. Propose mitigation options based on the assumptions and analysis:

- Revert to original encoding (one-hot): If the encoding change is the primary cause, reverting to one-hot encoding would address this issue.
- Oversample 'enterprise' tier in training: If the underrepresentation of 'enterprise' tier is causing the precision drop, addressing this imbalance during training can help.
- `sins(encoding)`: Use encoding methods that can handle skewness better to capture the unique patterns of the 'enterprise' tier.
- Feature engineering: Create new features that might help the model generalize better to the new serving distribution.

### 6. Recommend monitoring and rollback strategies:

- Monitor metrics: Keep track of the online precision metric to ensure the model's performance doesn't degrade further.
- Version control and A/B testing: Implement version control and A/B testing to compare the performance of different encoding methods and make informed decisions.

- Shift detection: Monitor distribution shifts to identify new issues early.
- Retry old encoding (canary release): As a temporary measure, test the old encoding alongside the new one in the production environment.
- Automated rollback: Implement an automated rollback system to reverse the encoding change if the online precision falls below a predetermined threshold.

#### 7. Validate assumptions:

- By implementing the proposed mitigation options, monitoring strategies, and rollback recommendations, we can validate or invalidate the assumptions made.
- If the precision improves after reverting the encoding change, it validates Assumption 1. Similarly, if oversampling the 'enterprise' tier in training or using an encoding method that handles skewness better improves the precision, it validates Assumption 2.

By following this

**Engineering counterpart**  
**3:46 AM - Responded in 0s**

What would you verify before shipping this to production?

## Agent response

3:46 AM - Responded in 17s

Before shipping the proposed changes to production, you should verify several aspects to ensure the validity, safety, and reliability of the updates. Here's a list of verifications and checks to perform:

### 1. Data Quality and Integrity:

- Ensure that the data used for retraining and evaluation is representative and of high quality.

- Validate that there are no data leaks or target leakage that could affect the model's performance.

- Verify that the training and evaluation datasets are independent and do not overlap.

### 2. Model Performance:

- Offline Evaluation:

- + Evaluate the retrained model's performance (if applicable) using the same metrics mentioned in the artifact (e.g., offline AUC on a holdout set) to ensure it's performing well on the training data.

- + Assess other relevant metrics, such as AUC-ROC, precision, recall, F1-score, and others, depending on your specific use case.

- Online Evaluation (A/B Testing):

- Before completely rolling back or implementing new encoding methods, perform A/B testing to compare the performance of the old and new models in the production environment.

- Use a representative sample of live traffic for the A/B test to ensure that the results are indicative of the overall production performance.

- Monitor metrics like online precision, as mentioned in the artifact, along with other relevant metrics, during the A/B test.

### 3. Model Fairness and Bias:

- Evaluate the model's fairness and bias, especially focusing on the 'enterprise' tier to ensure that the updates do not introduce or exacerbate fairness issues related to this underrepresented group.

- Analyze metrics like demographic parity, equal opportunity, and equalized odds, depending on your specific fairness concerns.

### 4. Model Interpretability:

- If possible, assess the interpretability of the model to understand how it makes predictions, especially focusing on the 'plan\_tier' feature and the 'enterprise' tier.

- Use techniques like SHAP, LIME, or feature importance to gain insights into the model's decision-making process.

### 5. System Integration and Compatibility:

- Verify that the updated model can seamlessly integrate with the existing production pipeline and infrastructure.

- Ensure that the model's input and output formats are compatible with the deployment system.

- Test the model's integration with other systems, such as data pipelines, monitoring tools, and alerting mechanisms.

### 6. Reliability and Scalability:

- Assess the model's reliability and scalability under varying loads and conditions.

- Perform load testing to ensure that the model can handle the expected production traffic and edge cases.

### 7. Rollback Plan:

- Have a well-defined and tested rollback plan in place in case the updates cause unforeseen issues or performance degradation.

- Ensure that you can quickly revert the changes if needed and have a plan for monitoring the model's performance post-rollback.

### 8. Documentation and Version Control:

- Document the changes made, the rationale behind them, and the verification steps taken.

- Implement version control for your models to track changes and enable easy rollbacks if necessary.

### 9. Stakeholder Communication:

- Communicate the changes, their expected impact, and the verification steps taken to relevant stakeholders, such as data scientists, engineers, and business users.

- Set clear expectations for the model's performance, monitoring, and feedback loops.

By verifying these aspects, you can ensure that the proposed changes are safe, reliable, and likely to improve the model's performance in production. Keep in mind that continuous monitoring and evaluation are essential to ensure that the model's performance remains stable and meets business objectives over time.

**Engineering counterpart**  
**3:46 AM - Responded in 0s**

What are the top risks, and what's the rollback plan if this doesn't work?

## Agent response

3:47 AM - Responded in 24s

Top Risks:

1. Performance Degradation: The primary risk is that the proposed changes might not improve or could even worsen the model's precision in production. This could lead to incorrect predictions and negatively impact the business outcomes associated with the churn prediction task.
2. Model Bias or Discrimination: The updates might introduce or exacerbate model bias, especially for the underrepresented 'enterprise' tier. This could result in unfair treatment of certain customer segments or non-compliance with relevant regulations or industry standards.
3. System Instability or Compatibility Issues: The changes might cause instability or incompatibility with existing systems, leading to downtime, errors, or other operational problems.
4. Data Leakage or Privacy Concerns: If not handled properly, the updates might introduce data leakage or privacy concerns, which could violate user privacy or regulations like GDPR, CCPA, or others.
5. Managing Expectations: Stakeholders might have different expectations about the potential improvements, and managing these expectations can be challenging if the results are not as expected.
6. Long-term Impact: The changes might have unintended consequences or side effects that only manifest over time, making it difficult to pinpoint the cause of any performance degradation or other issues.
7. Resource Waste: If the proposed changes do not yield the expected improvements, resources (time, effort, and computing power) spent on developing and testing the updates will be wasted.

Rollback Plan:

In case the proposed changes do not improve the model's performance or cause unforeseen issues, it's crucial to have a well-defined and tested rollback plan. Here's a step-by-step rollback plan:

### 1. Preparation:

Before implementing the changes, ensure that you have a complete backup of the existing production model and its associated artifacts (e.g., data, code, and configuration files).

Store the backup in a secure, easily accessible location, such as version control systems, cloud storage, or a dedicated backup server.

Document the rollback process and share it with relevant stakeholders.

### 2. Monitoring and Triggers:

Implement real-time monitoring and alerting mechanisms to detect performance degradation, bias, or other issues promptly.

Define clear thresholds or criteria for triggering a rollback, such as a significant drop in online precision or an increase in model bias.

Regularly review and analyze the model's performance and other relevant metrics to detect any degradation or concerns early.

### 3. Manual or Automated Rollback:

Manual Rollback:

- If a rollback is triggered, follow the documented rollback process to revert the changes.
- Restore the backed-up production model and its associated artifacts.
- Test the rolled-back model in a staging environment to ensure it's functioning correctly before deployment to production.

Automated Rollback:

- Implement an automated rollback system that can revert the changes based on predefined triggers or rules.
- Automated rollbacks should follow the same steps as manual rollbacks, including restoring the backed-up model, testing in a staging environment, and deploying to production.

### 4. Communication and Transparency:

Communicate the rollback to relevant stakeholders, explaining the reason for the rollback and the steps taken to minimize disruptions.

Maintain transparency throughout the rollback process, providing regular updates on the status and expected timeline.

### 5. Post-Rollback Evaluation:

After the rollback, closely monitor the model's performance and other relevant metrics to ensure that the production environment has stabilized.

Evaluate the causes of the performance degradation or other issues and document the lessons learned to inform future updates and improvements.

6. Investigation and Root Cause Analysis:

If the rollback is successful, conduct a thorough investigation to identify the root cause of the performance degradation or other

**Engineering counterpart**  
**3:47 AM - Responded in 0s**

Can you tighten this into concrete next steps with owners and timing?

## Agent response

3:47 AM - Responded in 19s

Here are concrete next steps with owners and estimated timelines to address the train/serve skew issue and mitigate the risks associated with the proposed changes.

### Week 1: Assessment, Planning, and Preparation

#### 1. Data Review and Validation:

Owner: Data Engineer/Specialist

Timeline: 2-3 days

Tasks:

- Validate the quality and integrity of the training and evaluation datasets.
- Ensure that there are no data leaks or target leakage.
- Confirm that the training and evaluation datasets are independent.

#### 2. Stakeholder Alignment and Expectation Management:

Owner: ML Engineer/MLOps Engineer

Timeline: 1-2 days

Tasks:

- Clearly communicate the proposed changes, their expected impact, and the rollback plan to relevant stakeholders.
- Manage stakeholder expectations and address any concerns or questions.

#### 3. Backup and Documentation:

Owner: MLOps Engineer

Timeline: 1 day

Tasks:

- Create a complete backup of the existing production model and its associated artifacts.
- Document the rollback process and share it with relevant stakeholders.

#### 4. Monitoring and Alerting Setup:

Owner: MLOps Engineer

Timeline: 2-3 days

Tasks:

- Implement real-time monitoring and alerting mechanisms for the model's performance, bias, and other relevant metrics.
- Define clear thresholds or criteria for triggering a rollback.

### Week 2: Model Evaluation and Testing

#### 1. Offline Model Evaluation:

Owner: ML Engineer/Data Scientist

Timeline: 3-4 days

Tasks:

- Retrain the model using the proposed changes (if applicable) and evaluate its performance on the training data.
- Assess other relevant metrics, such as AUC-ROC, precision, recall, F1-score, and others, depending on your specific use case.

#### 2. A/B Testing and Online Evaluation:

Owner: MLOps Engineer/ML Engineer

Timeline: 3-4 days

Tasks:

- Deploy the updated model alongside the existing production model for A/B testing.
- Monitor the performance of both models in the production environment using a representative sample of live traffic.
- Compare the metrics collected during the A/B test to evaluate the impact of the proposed changes.

### Week 3: Model Interpretation, Fairness, and System Integration

#### 1. Model Interpretability:

Owner: ML Engineer/Data Scientist

Timeline: 2-3 days

Tasks:

- Analyze the interpretability of the updated model, focusing on the 'plan\_tier' feature and the 'enterprise' tier.
- Use techniques like SHAP, LIME, or feature importance to gain insights into the model's decision-making process.

2. Model Fairness and Bias:

Owner: ML Engineer/Data Scientist

Timeline: 2-3 days

Tasks:

- Evaluate the fairness and bias of the updated model, especially focusing on the 'enterprise' tier.
- Analyze metrics like demographic parity, equal opportunity, and equalized odds, depending on your specific fairness concerns.

3. System Integration and Compatibility:

Owner: MLOps Engineer/Site Reliability Engineer (SRE)

Timeline: 3-4 days

Tasks:

- Verify that the updated model can seamlessly integrate with the existing production pipeline and infrastructure.
- Ensure that the model's input and output formats are compatible with the deployment system.
- Test the model's integration with other systems, such as data pipelines, monitoring tools, and alerting mechanisms

**Engineering counterpart****3:47 AM - Responded in 0s**

What's the one thing you'd want a second pair of eyes on?

**Agent response****3:48 AM - Responded in 24s**

The one thing I'd want a second pair of eyes on is the online evaluation (A/B testing) design and analysis. Here's why this is crucial:

1. Complexity and Nuances: Online evaluations can be more complex and nuanced than offline evaluations, especially when dealing with wuelu traffic and shifting distributions. A second pair of eyes can help spot potential issues, biases, or misinterpretations in the A/B testing design or results.
2. Stability and Confidence Intervals: A/B testing involves dealing with varying traffic, noise, and potential fluctuation in model performance. Ensuring that the results are statistically significant and taking into account confidence intervals is essential for making data-driven decisions. A second pair of eyes can help validate the statistical analysis and interpretation of the results.
3. Traffic and Traffic Shaping: To ensure a fair and representative comparison, it's essential to monitor and control traffic distribution during A/B testing. A second pair of eyes can help identify any traffic shaping issues, biases, or imbalances that could affect the results.
4. Evaluation Metrics and Trade-offs: The choice of evaluation metrics and how they are weighted can impact the interpretation of A/B testing results. A second pair of eyes can help ensure that the chosen metrics align with the business objectives and weigh the trade-offs appropriately.
5. interpretation and Contextualization: A/B testing results should be interpreted in the context of the business, the model's performance in other environments, and the specific use case. A second pair of eyes can help challenge assumptions, question potential biases, and provide a fresh perspective on the results.

To maximize the benefits of having a second pair of eyes, consider the following:

Choose an external reviewer who is not directly involved in the project or the A/B testing process to bring an independent perspective.

Provide the reviewer with all the relevant data, code, and documentation related to the A/B testing design, execution, and analysis.

Clearly communicate the business objectives, hypotheses, and expected outcomes of the A/B test to the reviewer.

Encourage the reviewer to ask questions, challenge assumptions, and share their thoughts freely.

By having a second pair of eyes review the online evaluation (A/B testing) design and analysis, you can increase the confidence and validity of the results, ensuring that the proposed changes are based on solid evidence and data.

**Engineering counterpart****3:48 AM - Responded in 0s**

Thanks - skew source and mitigation path are clearer now. Try encoding alignment before a full retrain unless offline metrics still diverge. Review complete - no further response needed.